CENTRO DE INVESTIGACION EN MATEMÁTICAS, A.C.

SOFTWARE PRODUCT LINES

Reporte Técnico de Investigación
que para obtener el grado de

Maestro en Ingeniería de Software

presenta

María Karen Cortés Verdín

Director:

Dr. Cuauhtémoc Lemus Olalde

Guanajuato, Gto. a 22 de julio de 2005.

CONTENTS

ABSTRACT

Software Product Lines has emerged as a new, promising engineering approach to resolve common problems in software development: reduced time to market, increased productivity, improved quality, managed complexity and customer satisfaction. Software Product Line Engineering or product Family Engineering encompasses, in addition to the traditional software engineering and management practices, additional challenges for those adopting this new approach. Therefore, despite the fact that there are a number of organizations actually operating under this new engineering approach, there are still a good number of areas of opportunities for research. Therefore, the purpose of this technical report is to review and investigate some of the main activities encompassed within Software Product Line Engineering, in order to identify research opportunities. An introduction to the main process frameworks for Software Product Line Engineering currently available is given. Next, a description of the activities so far investigated is presented: software product line scoping and measures, along with the research opportunities identified. Finally, conclusions are given.

1 Introduction

A Software Product Line is a "set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way"[1]. A software product line (or software product family) approach promotes planned and proactive reuse of core assets and architecture-centric development, achieving a substantial increment in product quality and a reduced time to market. Because of this, Software Product Line Engineering has received a lot of attention in recent years.

The keys to success for a software product line (SPL) effort are: exploring commonalities among products to proactively reuse software artifacts (core assets), encouraging architecture-centric development, and having a two-tiered organizational structure (core asset development and product development).

The definition of a framework for adopting, institutionalizing, managing and maintaining a software product line approach has been addressed by several organizations within the Software Engineering community. The Software Engineering Institute (SEI) at Carnegie Mellon University has developed his own framework (Framework for Software Product Line Practice[11]), while european organizations have decided to collaborate, integrating their own methods, processes and frameworks in a catalogue of methods supporting their own product line engineering process. In this way, they offer a variety of solutions for some areas or activities of SPL Engineering.

SPL Engineering is a young discipline. As such, there is plenty of opportunities for research. Therefore, the purpose of this report is to present the initial investigation done in the area of SPL Engineering such that opportunities of future research are identified. In order to this, the report is organized as follows:

The second section corresponds to an introduction to SPL as well as a description of the main approaches for Software Product Line Engineering.

The third section presents research opportunities so far identified. This mainly corresponds to the areas of product line scoping and measurement.

Finally, conclusions are presented.

2 Software Product Lines

A Software Product Line or Software Product Family was defined for the first time by David Parnas: “We consider a set of programs to constitute a family whenever it is worthwhile to study programs from the set by first studying the common properties of the set and then determining the special properties of the individual family members.”[2] In more recent years the Software Engineering Institute defined a software product line as a “set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way”[1]. A software product line or family approach promotes proactive and planned reuse, seeking to construct high quality products with an improvment in productivity, time to market and reduced product costs.

There have been several efforts, from different organizations, in trying to define a process or framework for software product line engineering (SPLE). Among them, the Software Engineering Institute at Carnegie Mellon University has developed a Framework for Software Product Line Practice. This framework will be described in further detail in section 2.1

Other efforts worth mentioning are those from european organizations, which, since 1995, have been cooperating in a series of projects from ITEA[7] (Information Technology for European Advancement). From these projects, the most outstanding are ESAPS[6] and CAFÉ[8]. They allowed to obtain a series of methods, processes, and work packages aimed at several activities or tasks within Product Family Engineering (PFE). These products have been integrated in a Catalogue of Methods and Processes for System-Family Engineering[20]. The framework for PFE will be described n section 2.2.

2.1 SEI’s Framework for Product Line Practice

SEI’s Product Line Practice Initiative[10] has developed the Framework for Software Product Line Practice[11]. SEI identifies three essential activities:

1. Core asset development. The goal of this activity is to establish the software product line production capability. Core assets are the basis for the production capability, among these assets there usually are: reusable software components, domain models, requirements, performance models, test plans, budgets, schedules, process descriptions and the architecture. The architecture is a key core asset for the production capability. Inputs to core asset development are: product constraints, styles, patterns and frameworks, production constraints, production strategy and inventory of preexisting assets. Outputs of this activity are: the product line scope, the core asset base and the production plan.

2. Product development. The development of products within the product line. The inputs to this activity are the product line scope, the core assets and their production plan and the specific product requirements.
3. Management. Two levels of management should be considered for the software product line (SPL) approach: organizational and technical. Both levels should be committed to the software product line approach in order for it to be successsful. Organizational management is defined as "the authority that is responsible for the ultimate success or failure of the product line effort."[1] Organizational management must set the proper organizational structure for the product line effort and determine a funding model that ensures core asset evolution. Technical management, on the other hand "oversees the core asset development and the product development activities by ensuring that the groups that build core assets and the groups that build products are engaged in the required activities, follow the processes defined for the product line, and collect data sufficient to track progress."[1]

The relationship among these essential activities is depicted in Fig. 2.1.

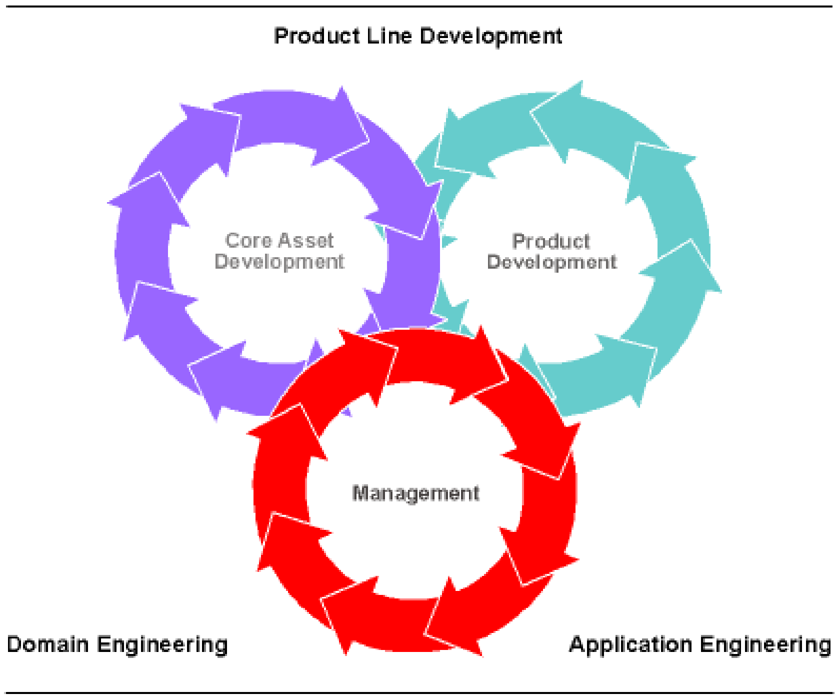


Fig. 2.1. Three essential activities for Software Product Lines taken from [24].

Each rotating circle represents one of the essential activities. The three activities are linked together in perpetual motion. This figure shows that all three activities are essential, are inextricably linked, can occur in any order, and are highly iterative. The rotating arrows indicate that core assets are used to develop products, and that revisions of existing core assets or new core assets evolve out of product development.

These essential activities are supported by specific practice areas which are categorized in: software engineering, technical management and organizational management. These areas are:

- Software engineering practice areas deal with “the application of the appropriate technology to create and evolve core assets and products.”[1]. The specific practice areas are:
    - Architecture Definition
    - Architecture Evaluation
    - Component Development
    - COTS utilization
    - Mining Existing Assets
    - Requirements Engineering
    - Software Systems Integration
    - Testing
    - Understanding Relevant Domains
- Technical management practice areas manage and support the software engineering practice areas.The specific practice areas are:
    - Configuration Management
    - Data Collection, Metrics and Tracking
    - Make/Buy/Mine/Comission Analysis
    - Process Definition
    - Scoping
    - Technical Planning
    - Technical Risk Management
    - Tool Support
- Organizational management practice areas enable and orchestrate software engineering and technical management practice areas. The specific practice areas are:
    - Building a Business Case
    - Customer Interface Management
    - Developing an Acquisition Strategy
    - Funding
    - Launching and Institutionalizing
    - Market Analysis
    - Operations
    - Organizational Planning
    - Organizational Risk Management

- Structuring the Organization
- Technology Forecasting
- Training

2.2 ESAPS and CAFÉ Product Family Engineering Process

ESAPS[6] (Engineering Software Architectures, Processes and Platforms for System-Families), as mentioned before, was developed through ITEA framework. The purpose of ESAPS was to provide the technologies to help companies in successfully adopting a product family approach. It builds upon the results of two previous projects: PRAISE[9] and ARES[4]. PRAISE was focused on domain and application engineering while ARES was architecture-centric.

CAFÉ[8] (Concepts to Application in System-Family Engineering), on the other hand, extended the work done in ESAPS, integrating the separate concepts of ESAPS in a unified whole covering product family's entire life cycle[4]. The focus of CAFÉ was the introduction of a product family approach in an organization.

The process for Product Family Engineering is defined as follows:

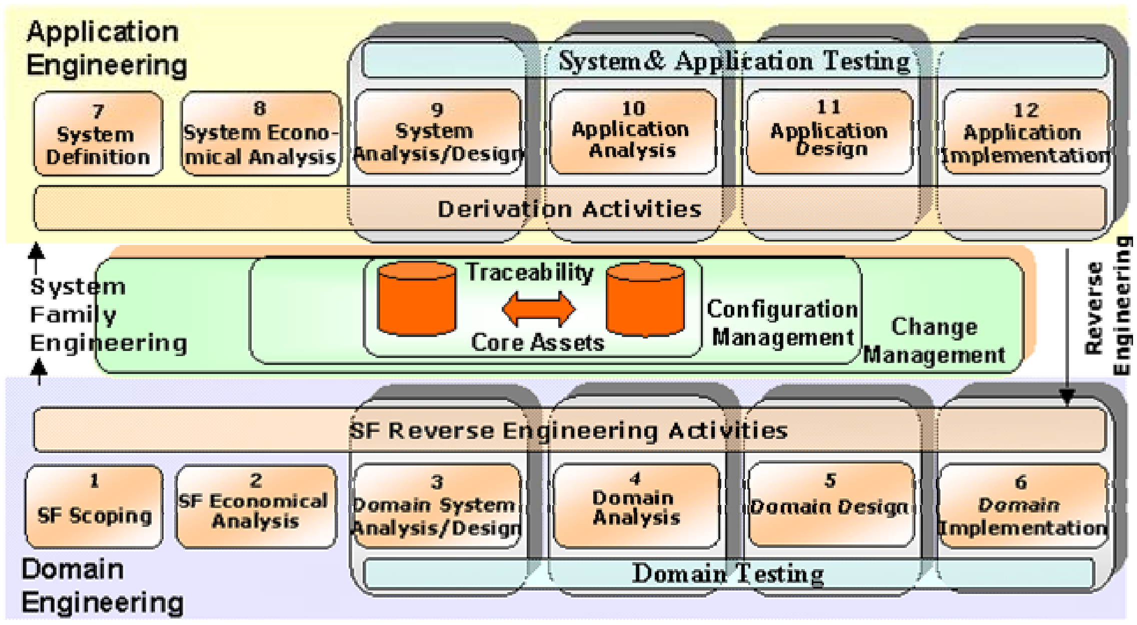


Fig. 2.2. The Product Family Engineering Process taken from [20].

As shown by Fig 2.2. the PFE process comprises three big areas: application engineering, domain engineering and system family engineering. Application engineering refers to the process of developing products within a product family whereas domain engineering is the process of developing reusable assets (or core assets) that serve as the basis for developing the individual products in a family. System Family engineering encompasses both, application and domain engineering, that is to say, the whole PFE approach. For each one of the activities in the process, the activities of the PFE process and related outputs obtained by projects' participants are[20]:

1. Software Family Scoping, is the process of identifying and bounding the focus of development for reuse in product line development. ESAPS participants distinguished three kinds of scoping[4]:

    - Product line scoping. "Is the process of systematically developing a Product Portfolio Definition. A Product Portfolio Definition is in turn defined as a description of the specific requirements and the individual products that should be part of the product line."[16]
    - Domain scoping. "Is the process of identifying appropriate boundaries for a domain which are relevant for implementing systems in the product line." [16]
    - Asset scoping. "Is the process of identifying the various elements that should be made reusable." [16]

    methods addressing this activity will be described in section 3.1.2.

2. System Family Economical Analysis. Adopting a SPL approach involves significant investments for the organization. Therefore, there is a need for an economic model to determine the costs and benefits of such an approach. Among the methods for this activity are: Fraunhofer IESE PuLSE-ECO, Ivorium's lightweight approach and Product-Line Action Plan from the European Software Institute (ESI).

3. Domain System Analysis/Design. Domain analysis is the process by which information used in developing software systems within the domain is identified, captured, and organized with the purpose of making it reusable (to create assets) when building new products. In this way, domain analysis serves to identify commonalities and variabilities in requirements and capture decisions on the ranges and interdependencies of variabilities. Domain design is the process of developing a design model from the products of domain analysis and the knowledge gained from the study of software requirement/design reuse and generic architectures.[21]

4. Domain Analysis. Outcomes for this activity are related to domain engineering methods and techniques. Among them, Conceptual domain analysis is the process which is specifically covered although there is no general notation used for it. FODA[15], UML[23], and mind maps are some of the methods and techniques that can be employed.

5. Domain Design. Domain engineering methods are also used. Specific process frameworks that address domain design are those from Siemens (MoVE), ALCATEL (SPLIT), and Fraunhofer IESE (PuLSE).
6. Domain Implementation. It is the process of implementing reusable components (core assets) based on the domain model and generic architecture. Using the domain knowledge gathered during domain analysis, and the generic architecture developed during domain design, domain engineers acquire and, where necessary, create reusable assets. Creation, management, and maintenance of a repository of reusable assets are also important parts of domain implementation.[22]
7. System Definition. It is the first activity in product development within a product family. Specialized requirements for the product must be elicited and specified, ensuring compatibility with the product family scope.
8. System Economical Analysis. Outcomes obtained are: Process for reverse architecting, Method for Aspect-driven development, Traceability methods, Scoping methods, and the Software Product Family Engineering process frameworks developed by each one of the projects' participants.
9. Application Analysis. Methods for the application analysis phase, such as: Model-driven requirements engineering, Natural language techniques for Product Families Software requirements, Feature trees, Development by means of scenarios, Feature analysis, and Asset management.
10. Application Design. Methods for the application design phase, such as: Architecture recovery, Software architecture assessment, Architectural mismatches analysis, Architecture evolution, Platform Independent Modeling, Platform Specific Modeling, and configuration and derivation of product architectures.
11. Application Implementation. This activity refers to the construction or implementation of a member of the family. It encompasses the instantiation of the reference architecture and the product family model, the creation or reuse of core assets and the validation of the resulting application or product. Among the related methods are: Agile product line engineering, Code generation, Behavior modeling, Interface evolution, Transition process, Design management, and Configuration management process.

3 Research Opportunities

As stated in the introduction of this technical report, the main objective of the present work is to identify research opportunities in the field of Software Product Line Engineering. The purpose of this section is to present research areas in which such opportunities exist. First, a review of Software Product Line Scoping status is given. This is one of the initial activities to be performed for adopting a SPL approach. SPL scoping has a strong relationship to business objectives and drives the whole asset and product development processes.

Next, a description of Measures for SPL is given. The information here presented is basically based on the SEI's Framework for Software Product Line Practice and on the work done by Zubrow et al[25]. Investigation still remains to be done in ESAPS and CAFÉ projects.

3.1 Software Product Line Scoping

Basically, the adoption of a SPL approach, includes the following[2] major phases:

- Determine stakeholders
- Create business cases
- Create adoption plan
- Launch and institutionalize

As the first step, the stakeholders of the SPL effort must be determined. Stakeholders will have different interests in product line adoption, therefore business cases will depend on such interests. The business cases will help stakeholders in achieving the product line goals. Next, an adoption plan must be elaborated. The adoption plan establishes goals, strategies and acivities to be perfomed to make the transition to a product line approach. This plan serves to decide about the product line adoption. If adoption is chosen, the product line is launched. It is important that once launched, the product line effort is institutionalized: managers and staff should consider it as part of their working culture.[2]

A key activity or practice area in product line planning is scoping.[5] Product line scoping is the activity that "bounds a system or set of systems by defining those behaviors or aspects that are *in* and those behaviors that are *out*"[1] the product line. In other words, scoping helps to identify those products that will be within the product line in such a way that the product line is profitable.

Product line scoping helps to clarify which requirements will be implemented in the core assets and which in the products, focusing the reuse investment where it will pay. When the scope is too large, the core assets will be too general to be properly used. When the scope is too small, there won't be a market for the product line. Therefore, the importance of product line scoping.

Scoping identifies commonalities and variabilities among members, it is essential to determine whether a proposed system can be built within the product line and from product line assets[1]. A product line scope, should derive from the product line objectives. Product line objectives themselves, are built upon a business case of the organization. Due to this, the product line scope pervades along the product line effort, becoming in this way, a valuable core asset.

In this section, current approaches to Software Product Line Scoping are described. Several approaches can be identified: the one proposed by the SEI's Framework for Product Line Practice and those resulting from european projects ESAPS and CAFÉ.

3.1.1 SEI's Framework for Product Line Practice

SEI's Product Line Practice Initiative[10] has developed the Framework for Software Product Line Practice. As mentioned in the previous section, SEI identifies three essential activities: core asset development, product development and management. These essential activities are supported by specific practice areas (software engineering, technical management, and organizational management). Scoping is an activity belonging to technical management practice area.

According to this framework, product line scoping involves the following specific practices:

- Examining existing products. To conduct a study of existing products to identify commonality and types of differences across a potential product line.
- Conducting a workshop to understand product line goals and products. To gather potential product line stakeholders and establish the direction for the product line.
- Context diagramming. Developing a context diagram allows to place the product line in the context of other systems and of product users. This eases the identification of elements affecting and affected by the product line.
- Developing an attribute/product matrix. To develop an attribute/product matrix to sort, in order of priority, the attributes that differentiate the products in the product line.
- Developing product line scenarios. Scenarios are very useful in defining a product line's scope since they identify user or system interactions with product line products.

In addition to these specific practices, SEI's framework does not propose a particular method for product line scoping. However, it suggests domain engineering methods which can be applied to domain scoping, like Organization Domain Engineering (ODM)[14] and Feature-Oriented Domain Analysis (FODA).

3.1.2 ESAPS and CAFÉ

As mentioned before, ESAPS participants distinguished three kinds of scoping:

- Product line scoping.
- Domain scoping.
- Asset scoping.

ESAPS only covered domain and asset scoping. Strategies for domain scoping are related to domain analysis techniques whereas strategies for asset scoping are related to feature analysys. Approaches for product line scoping pertain to market science and only two organizations among the participants of ESAPS provide some way to address this kind of scoping. These organizations are Fraunhofer IESE and Siemens.

CAFÉ outcomes of this project related to scoping are the following:

- Scoping in the presence of Multiple Domains and Product Populations. A product population is a product family with great diversity, or a set of product families that share a common (sub)domain. Multiple domains refer to converging domains. That is to say, domains that can converge on various levels[12]: technical or realization, functionality or application, and/or marketing. In this situation, the presence of more stakeholders from different domains is needed. This can be a problem due to the fact that because the stakeholders come from different domains, they speak different languages. So, the purpose of this method is a scoping approach based on user scenarios, that enables the cooperation of stakeholders with different expertise and knowledge.The outcome is a rough product family scope from which more formal scoping approaches can start. This method was developed by Philips.
- PuLSE-ECO. After CAFÉ finished, this method has continued to evolve, and nowadays the 2.0 version is available. The method belongs to a complete Software Product Line Engineering process, known as PuLSE^{TM} (Product Line Software Engineering). This method, and the process, were developed by Fraunhofer IESE[13] and is described in more detail in section 2.3.
- Siemens. The approach proposed by Siemens for scoping is based on PuLSE-ECO V2.0 with its own scoping decision information model. This approach is called MoVE (Model-based Value Engineering). In MoVE a decision consists of two parts: the decision maker and the decision problem.

The decision maker has a value system consisting of a number of raw objectives, which in turn are ordered according to the decision maker's preferences. A decision problem consists of several classes of decision elements[16]:

- Requirements.
- Importance measure.
- Alternative product definition.
- Performance measure.
- Objective function

MoVE supports decision making by assessing different configurations of requirements, features or products in a product line. Each product is defined by a set of requirements and a set of realization concepts that realize the requirements. One or more configurations of different realization alternatives fulfill each requirement. Each configuration is assessed in relation to cost, benefit, synergies and other parameters, and the resulting assessment supports the choice of a certain requirement, feature or product within the product line.

During product line scoping, the requirements for each candidate product are prioritized. The requirements should be allocated to features and their realization concepts. Then, this concepts should be parameterized and configurations fullfilling the specific product requirements should be identified. The configurations should then be assessed acording to the importance measure. This assessment is the basis for deciding if the product should belong to the product line.

For domain scoping, MoVE assesses and measures the goodness of realization concepts and configurations, determining in this way, the number of features and products in a domain.

In asset scoping, the decision refers to whether an element such as a feature, function or component, should belong to the product line, or should be application-specific. This is done by calculating the costs of making this element reusable and the benefits obtained by this reusable element. Then, the criteria for an assessment of this decision have to be determined.

It is important to note that the packaging of requirements (the allocation of requirements to a product line member) is a major problem, that in MoVE is addressed by identifying market segments. A market segment maximizes homogeneity of the requirements. In this way, a product definition is obtained for each segment.

- Specification for a Product Family Scoping Approach and its integration with a Tool Workbench. The method was developed by Ivorium[17] and is focused on lightweight methods. This approach looks for the determination, in advance, with certain level of certainty, of the return on investment that adopting a product family approach would bring to the organization. In order to this, the approach seeks to integrate ROI analysis with scoping analysis since they feed each other. The dimensions of the approach are[17]:
    - Enterprise goals: goal definition as the articulation of what the enterprise hopes to achieve by adopting a product family approach.
    - Product Family definition: The concept of a product map is used and augmented to define what will belong to the product family.
    - Family scoping and ROI: by leveraging a concrete goal decomposition and product maps, a product family scope and ROI can be computed.
    - Risk evaluation: Given the scoping and ROI results, an analysis is done to evaluate risks.

  Ivorium identifies the following scoping types:

    1. Top down. Analysis in which different products must be evaluated to determine the best scope for the product platform and the best products to be actually developed (best meaning highest ROI given the organization's goals).
    2. Top down with competition. It is the same as top down with the addition of one or more base products from which the reusable platform can (and will) originate.
    3. Bottom-up. When the organization does not have a clear product strategy and management wants to adopt a product family approach, the most valuable direction in terms of marketing and engineering must be chosen.
    4. Product policy. It is a top-down effort based on actual construction with an orientation towards structuring a product policy.

  This approach allows an organization to:

    - Concrete definition of the product family options are in terms of its possible features and products.
    - Definition of quantitative goals, and their alignment to the organization's goals.

    - Scoping of the best product family feature and/or product platform, easing in this way product family adoption since the potential efficiencies are defined in terms relevant to the organization’s projects and in terms of goals and valuations.

- Scoping Software Product Lines for the Business Context – Agility, developed by Nokia[18]. A software product line approach increases complexity and therefore it is managed by adding more formal procedures. Considering that nowadays organizations are operating in a dynamic market, the bureaucracy imposed by SPL management methods can slow the organization’s reaction to such an environment. Nokia’s approach takes these factors into consideration and proposes a model for software product line scoping such that it allows an organization to choose a software development approach in accordance to this kind of environment. This model includes the concept of agility within the context of software product line and provides an insight on how agile methods can be related to a software product line approach. The need for agility comes from the chosen business strategy. The model consists of two dimension or axes, as shown in Fig. 3.1. The vertical axis can be named as “Size & Complexity of the Software” and it corresponds to the challenge that managing software assets is putting to the organization. The horizontal axis corresponds to the business challenge that the organization is facing. This axis can be named “Volatility of the business environment”. Reasons for this volatility can be immature technology, and the competitive situation, for example. The two dimensions are measured by three scales. For business volatility, organizations can be operating in low volatility, medium volatility or high volatility environments. In the case of size and complexity of software, organizations can be classified in small size & complexity, medium size & complexity or large size & complexity. These measures help to determine an organization’s current situation and the software development approach to be chosen in order for it to adapt to its business environment. The traditional software development approach is recommended when an organization is heading towards large size & complexity of the software whereas agile methods are better towards high volatility of the business environment. Therefore, a combination of the two approaches is possible depending on the organization’s position in relation of both dimensions.

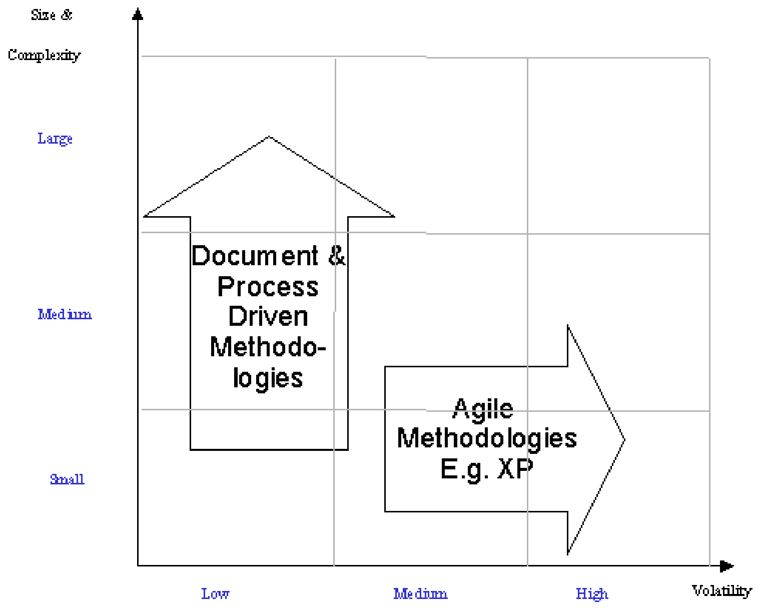

Size &
Complexity
Large
Medium
Small
Document & Process Driven Methodo-logies
Agile Methodologies E.g. XP
Low
Medium
High
Volatility

Fig. 3.1 Software development approaches and their applicability taken from.[8]

3.1.3 PuLSE-ECO V2.0

This approach, developed at the IESE, had as its main drivers or requirements[5]:

1. A base on products to be built rather than on.
2. To provide basis for communication among stakeholders.
3. Coverage of product line scoping activities: analysis of product line potential (risks and benefits) and identification of specific reusable assets.
4. The evaluation of the product line potential is performed on the level of the business domain as well as on the level of the contributing technical subdomains. The output of the asset scoping activity should be useful to identify reusable assets.
5. To be repeatable.

There are three main components in PuLSE-ECO V2.0: product line mapping, domain potential assessment and reuse infrastructure scoping.

Product line mapping (PLM) provides the means for communication along the process. It is a high level domain analysis which is the basis for the remainder of the process. Based on information from domain experts, the planned portfolio, product plans, and existing systems, PLM derives a standardized description of the product line using features as the common language for describing products as well as domains. Features correspond to functionality and therefore, non-functional requirementos or qualitiy attributes are not considered since they cannot be scoped from components[5].

Domain Potential Assssment (DPA) deals with the identification of risks and benefits pertaining to he domains considered in the product line. This component is based on process capability assessments (CMM, Bootstrap and ISO 15504). There are four process assessment concepts and DPA maps to each one of them:

Process Assessment Concept	**Domain Potential Assessment Concept**
A standardzed process for performing assessments is used.	A variant of the FAME-process is used for performing the assessment.
Process framework + process existence indicators.	The product line mapping method is usedfor developing a reference description of the domains relevant to the product line.
Stabdardized capability indicators are provided.	Assesment indicators have been specifically developed for this approach.
Rating Scheme.	A specialized rating scheme was developed along with the capability indicators.

Table 2.1 Mapping of Assessment Concepts taken from [5] .

FAME-process is a variant of the ISO 15504 adapted for the DPA. It uses a new evaluation framework which invoves the following key criteria[5]:

Viablity dimensions:
- Maturity
- Sability
- Resource constraints
- Organizatioal constraints

Benefit dimensions:
- Market potential – external
- Market potential – internal
- Commonality and variability
- Coupling and cohesion
- Existing assets

Reuse Infrastructure Scoping (RIS) determines which assets develop to reuse and which assets develop as specific to the product. RIS builds on the PLM since the latter obtained a product line breakdown structure in terms of domains and features. And RIS builds on the DPA because it has properly assessed the reuse potential. RIS is based on a quantitative analysis of the benefits that can be expected from reuse. Therefore, the combination of all the features that provide benefit constitutes the asset scope. RIS includes an operationalization of the business ojectives based on the Goal-Question-Metric(GQM) approach[5], and the construction of the corresponding model(s).

3.2 Measures for Software Product Lines

The responsibilities associated with each managerial role within a SPL approach, help to identify goals and issues to be addressed with information derived from software measures. These responsibilities are depicted by the following figure:

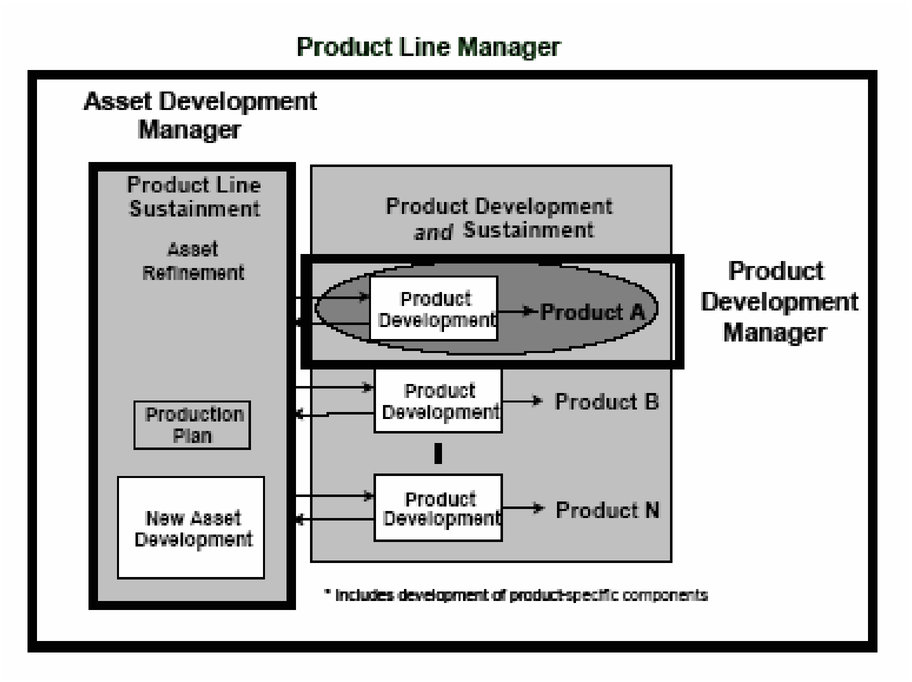


Figure 3.2 Product Line Management Roles taken from[25].

The product line manager is responsible for the overall business enterprise and is concerned with the entire set of past, current, and future products within the product line. The product line manager has the need to demonstrate the benefits associated with the SPL approach.

The asset development manager is responsible for the core assets and associated infrastructure. The asset development manager needs to provide high quality assets on a timely basis to product development.

The product development manager is responsible for a product within the SPL and is concerned with providing high quality products, on time and within budget, while conforming to SPL processes and contributing to the achievement of the SPL goals.

According to Zubrow[26] aspects related to a SPL approach that should be informed with software measurement and analysis are:

- The decision to adopt a SPL approach.
- The ongoing operation and the overall performance of the SPL.

A fundamental input to the decision to adopt is the economy of scope that might be realized. Economy of scope is “the extent an organization can leverage commonality across its software products to reduce costs while increasing the variety of products produced and supported.”[27] The organization’s investment in a SPL approach is large relative to traditional software development, therefore it needs to know the point on which there will be a return on its investment. This area is not well established or validated in terms of measurement and analysis. Some work has been done by participants of ESAPS and CAFÉ projects[20], but research is still on its way.

The ongoing operation and overall performance aspects are closely related to the following dimensions:

- Performance of development projects in relation to cost, schedule and quality objectives as compared to traditional software development.
- Compliance. The extent to which development projects utilize the processes, practices and standards designed to leverage and reuse large-grained, common assets.
- Effectiveness. The extent to which the SPL meets its goals and those of the organization.

The purpose of this section is to describe current approaches to Measures for Software Product Line. So far, only the SEI’s Framework for Product Line Practice approach has been investigated as well as the work done by Zubrow et al[25]. The related results are presented in the following paragraphs.

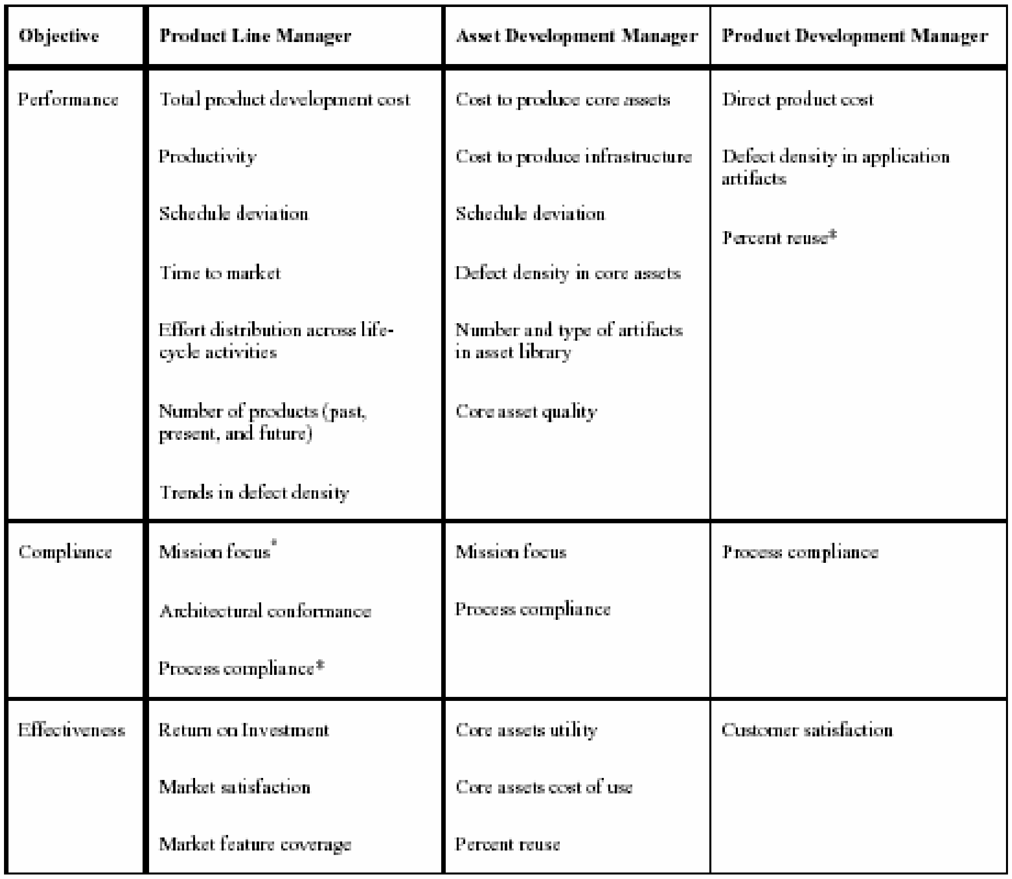
Within the SEI’s Framework, Data Collection, Metrics and Tracking is a specific practice area under the category of Technical Management. This practice area recognizes the need of measures for the adoption and the ongoing operation and performance of a SPL approach. The specific practices are:

- Choosing metrics. SEI’s framework is not specific on how to determine the metrics. It suggests the GQM approach, as well as the one proposed by Zubrow[26].

- Collecting data. Different types of metrics require different data collection techniques, among the most common techniques are:
  - Direct measurement of observable attributes of a process or product.
  - Indirect measurement of objective attributes.
  - The derivation of implicit attribute measures as computations from other measures.
- Reuse metrics. Reuse is a strategy for achieving the SPL goals, not a goal itself. However, it is useful to have metrics of what the reuse is giving to the organizations in terms, for example, of costs and return on investment.

It is important to note that the organization's maturity in data collection and measurement practices is a key aspect in the success of this practice area.

In relation to Zubrow et al[25] work, propose the following measures:



Objective	Product Line Manager	Asset Development Manager	Product Development Manager
Performance	Total product development cost	Cost to produce core assets	Direct product cost
	Productivity	Cost to produce infrastructure	Defect density in application artifacts
	Schedule deviation	Schedule deviation	Percent reuse*
	Time to market	Defect density in core assets	
	Effort distribution across life-cycle activities	Number and type of artifacts in asset library	
	Number of products (past, present, and future)	Core asset quality	
	Trends in defect density		
Compliance	Mission focus*	Mission focus	Process compliance
	Architectural conformance	Process compliance	
	Process compliance*		
Effectiveness	Return on Investment	Core assets utility	Customer satisfaction
	Market satisfaction	Core assets cost of use	
	Market feature coverage	Percent reuse	

Table 3.1 Product Line Indicators and Measures taken from[25].

The authors explain that this measures set is not an exhaustive one, and should therefore be taken as an initial proposal, leaving to the organization the evolution of those measures that closely fulfill their needs for its own SPL approach. A brief description of each one of the measures is given in the following paragraphs.

- Measures for Software Product Line Management.

Many of these measures are derived from enterprise and project management:

- Total product development cost. “It measures the engineering costs incurred by the product line organization to create new software products.”[25] This measure includes both, the direct product development cost and the prorated share of the asset development cost.
- Productivity. It is the ratio of the amount of product or ouput relative to the resources consumed to produce it.”[25] The output can be measured as the number of products fielded, number of features, LOC or function points. The resources are expressed as effort expended. Over time, the productivity should increase as the investment in core assets and infrastructure is spread across an increasing number of fielded products, and as the cost of products development decreases (due to the use of core assets and infrastructure).
- Schedule deviation. This measure consists of the sum of the variance of all product schedules. As the product line matures, it should be able to meet its schedules in a more reliable fashion.
- Time to market. This measure represents an organization’s capability to deliver products and features faster. This measure is based on the functionality delivered by the projects and on the projects’ duration.
- Number of products. This measure characterizes the scope of the SPL and its contribution to return on investment.
- Trends in defect density. Defects in delivered products reflect their quality. All defects in delivered products should be tracked in order to determine if quality is improving. This measure is calculated as the ratio of defects relative to the size of the product.
- Mission focus. This measure evaluates “the degree to which the products produced by the product line organization fit within its defined scope.”[25] Since the product line scope is defined according to the organization’s mission and objectives, this measure helps to track mission fulfillment.

    - Architectural conformance. This measure helps to preserve the integrity of the product line and facilitates an efficient use of the core assets in product development. For evaluating architectual conformance, measures of software architecture should be used. So far, measures in this area are just emerging.
    - Process compliance. “Captures the degree to which products are produced using the product line process.”[25] If a software quality assurance function is in place within the organization, data for this measure could come from process audits performed by this function.
    - Return on Investment. It is the ratio of estimated savings for each dollar invested. The challenge for this measure is to clearly and objectively estimate the savings on investment, the costs to transition to a SPL approach, and the costs to develop products using the SPL approach.
    - Market satisfaction. It is the customer satisfaction when purchasing products. This measure is generally obtained through a survey.
    - Market feature coverage. It is the extent to which features in the product cover those features related to the target market. The goal of this measure is the identification of features relevant to the market.
- Measures for Asset Development Management:
    - Cost (Effort) to produce Core Assets. These costs are similar to those in typical software development. These measures should track the development of software components as well as non-code assets such as requirements, architectures, and user documentation.
    - Asset development Schedule deviation. Core assets’ availability is extremely important for the operation of the entire product line. Therefore, asset development should be tracked against its planned schedule. Schedule deviation is computed as the duration between the planned and delivered dates, where planned dates are based on projected needs of product development projects.
    - Defect Density of Core Assets. Since the quality of the core assets will influence the quality of the final product, it is important to measure the quality of the core assets as well as the capabilities they offer. To obtain this measure, the defects reported are divided by the size of the corresponding core asset.
    - Core Asset Quality. As SPL have additional quality requirements to those of a single product, so do core assets. Core assets must comply various standards, specifications, architecture and there may be quality attributes to fulfill as well. The practical application and validation of measures in this area is unknown.

- Number and Type of Assets Available. This measure helps the asset development manager to monitor the development of core assets. This measure helps also to know about the growth of the SPL reusable base.
- Process compliance. This measure captures the degree to which core assets and tools are being produced in accordance with the organizational product line standards.
- Core Assets Utility. Core assets should provide value to product development. This value or utility gives an insight into the extent to which core assets are satisfying the goals of the SPL as well as helps determine whether the assets in the asset library are actually used to create products. One approach to the measure core asset utility is the effect their use has on product development project performance. This can be measured in terms or cycle time, costs and quality. If the data required for this measure are not available, percent reuse could be used instead. Or a count of the number of used of each core assert divided by the number of products in the product line can give an insight of which assets are more commonly used.
- Core Assets Cost-of-Use. Thos measure corresponds to the costs incurred by product development projects in order to use core assets effectively. If the core assets are too expensive to use, product development will seek other ways to fulfill product requirements. Typically, these costs are associated with:
    - Identifying appropriate core assets
    - Understanding now to apply and adapt the core assets
    - Integrating and testing

- Measures for Asset Development Management:

Measures for this area of management are the same as for traditional software development projects plus only a few specific measures for product lines. It must be taken into account that the setting for application project is the source of much of the data on performance of the product line, therefore, there should be a measurement system that provides for the collection of the needed data.

- Direct product cost. These costs should include all direct labor costs incurred while producing the product. Product development management should monitor the relative proportion of project costs going to produce application-specific code. Over time, it is expected that these costs will decline as the proportion of the core assets integrated into the products will increase and the quality of such core assets improves.

- Defect Density of Application-Specific Code. To compute this measure, defects reported by customers are divided by the size of the application-specific code in the product.
- Process compliance. Product development must comply with processes associated with the operation of the product line. This measure captures the degree to which product development complies such processes.
- Percent reuse. Tracking this measure is important sinc reuse is the principal strategy for product line success. Failure to achieve planned reuse levels may signal performance trouble for product development and may also trigger a causal analysis by core asset management. This measure can be computed as the ratio of the size of core assets used in relation to the size of new and modified application artifacts plus the size of core assets used
- Customer satisfaction. It is important to know how well products satisfy customers' needs. This information can be obtained via a customer satisfaction survey. These data should be retained and analyzed cumulatively as more products are produced.

Zubrow et al state that the utility of the above measures still remains to be validated. In addition, SPL organizations demand ways to assess the scope, features, variants, and software architecture from a business perspective.

3.3 Opportunities

The research opportunities identified in relation to Product Line scooping and measures are the following:

1. Product Line scope assessment
2. Product Line scope tracking and management along the SPL engineering process.
3. Architectural conformance of products with respect to the Product Line Architecture.
4. Measures for Product Line Software Architecture cost of use and utility (effectiveness).
5. Product Line evolution.
6. Product Line requirements engineering and scoping management.
7. Integration of SPL approach with upfront processes, such as marketing, requirements management and portfolio management.

8. Establishing a SPL engineering measurement program.
9. Implementation of a SPL measurement program that covers adoption and managing practice areas:
    a. Organization and support practices.
    b. Practices that balance platform versus client interests.
    c. Requirements engineering practices.
    d. Architectural practices.
10. Process for SPL architecture development .
11. Measurements for the transition process to SPL.
12. SPL measurement program for small and medium enterprises.

4 Conclusions

This technical report presented the research done so far in the area of Software Product Line Engineering. SPL Engineering is a relatively new field and its promises of reduced costs and time to market as well as improved productivity and quality have made organizations turn to it as an approach for software development.

Most important frameworks for SPL approach were presented: SEI's Framework for Product Line Practice and ESAPS & CAFÉ Product Family Engineering Process. Although different in their approach to SPL Engineering, both frameworks address important SPL practices and give or propose solutions at different levels for them.

The main activities practices so far investigated and presented in this work were Product Line Scoping and Measures. Approaches and methods from SEI's framework and ESAPS & CAFÉ PFE were presented. This investigation is just in its initial phases, there remains more investigation to be done and the research opportunities here presented must still be elaborated and validated. Among the areas still to be investigated are:

Architecture definition and evaluation

Product Family process frameworks such as SPLIT, PuLSE, and QUEST

Asset mining

Feature oriented engineering

Variation mechanisms

Reuse in the context of SPL

Risk management

SPL launching and institutionalizing

5 References

1. P. Clements, and L. Northrop, *Software Product Lines: Practices and Patterns*, Addison-Wesley, USA, 2001
2. G. Blöcke, J. Bermejo Muñoz, P.Krauber, C.Krueger, J.C. Sampaio, F. van der Linden, Linda M.Northrop, M. Stark, D.M. Weiss, "Adopting and institutionalizing a Product Line Culture", Proceedings of the 5th International Workshop on Product Family Engineering, 2003.
3. C., Krueger, Eliminating the Adoption Barrier, IEEE Software, vol. 19, no. 4, July/August 2002, pp.28-31.
4. F. van der Linden, "Software Product Families in Europe: The Esaps & Café Projects", IEEE Software, vol. 19, no. 4, July/August 2002, pp. 41-49.
5. K. Schmid, "A comprehensive Product Line Scoping Approach and Its Validation," IEEE International Conference on Software Engineering, ICSE'02, ACM, USA, 2002.
6. http://www.esi.es/en/Projects/esaps/esaps.html
7. http://www.itea-office.org
8. http://www.esi.es/en/Projects/Cafe/cafe.html
9. http://www.esi.es/en/Projects/Praise/praiseProject.html
10. http://www.sei.cmu.edu/productlines/plp_init.html
11. http://www.sei.cmu.edu/productlines/framework.html
12. http://www.esi.es/en/Projects/Families/E1.4b-Method-Catalogue/CAFE/Details1-v0.1.html
13. http://www.iese.fhg.de/
14. http://www.sei.cmu.edu/str/descriptions/odm_body.html
15. http://www.sei.cmu.edu/domain-engineering/FODA.html
16. J. Bosch, M. Jaring, S. Johnsson, K. Schmid, S. Thiel, B. Thomé, S. Trosch; K. Schmid (Ed.). Task 1.2: Domain Analysis: Consortium-wide deliverable on Scoping. Deliverable of ESAPS project, Eureka ∑! 2023 Programme, ITEA Project 99005, 2001.
17. J. M. DeBaud, X. Vaisson; J. M. DeBaud (Ed.). Task 1.2: Specification for a Product Famly Scoping Approach and its integration in a Tool Workbench: Consortium-wide deliverable on Product Line Scoping. Deliverable of CAFÉ project, Eureka ∑! 2023 Programme, ITEA Project ip00004, 2003.
18. T. KähKönen; J. M. DeBaud (Ed.). Task 1.2: Scoping Software Product Lines for the Business Context – Agility: Consortium-wide deliverable on Product Line Scoping. Deliverable of CAFÉ project, Eureka ∑! 2023 Programme, ITEA Project 99005, 2003.
19. D. L. Parnas, "On the Design and Development Of Program Families," IEEE Trans. Software Eng., vol. SE2, no. 1, Mar. 1976, pp. 1-9.
20. http://www.esi.es/en/Projects/Families/E1.4b-Method-Catalogue/Start_SFE_Catalogue.htm

21. http://www.sei.cmu.edu/domain-engineering/domain_design.html
22. http://www.sei.cmu.edu/domain-engineering/domain_imp.html
23. http://www.uml.org
24. http://www.sei.cmu.edu/productlines/frame_report/PL.essential.act.htm
25. D. Zubrow, G. Chastek, *Measures for Software Product Lines* (CMU/SEI-2003-TN.031). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.
26. D. Zubrow, *Measures for Software Product Lines: A White paper for the Office of the Undersecretary of Defense, Science and Technology, Software Engineering*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001.
27. J. Whitey, *Investment Analysis of Software Assets for Product Lines*, (CMU/SEI-96-TR-010, ADA 315653). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996.